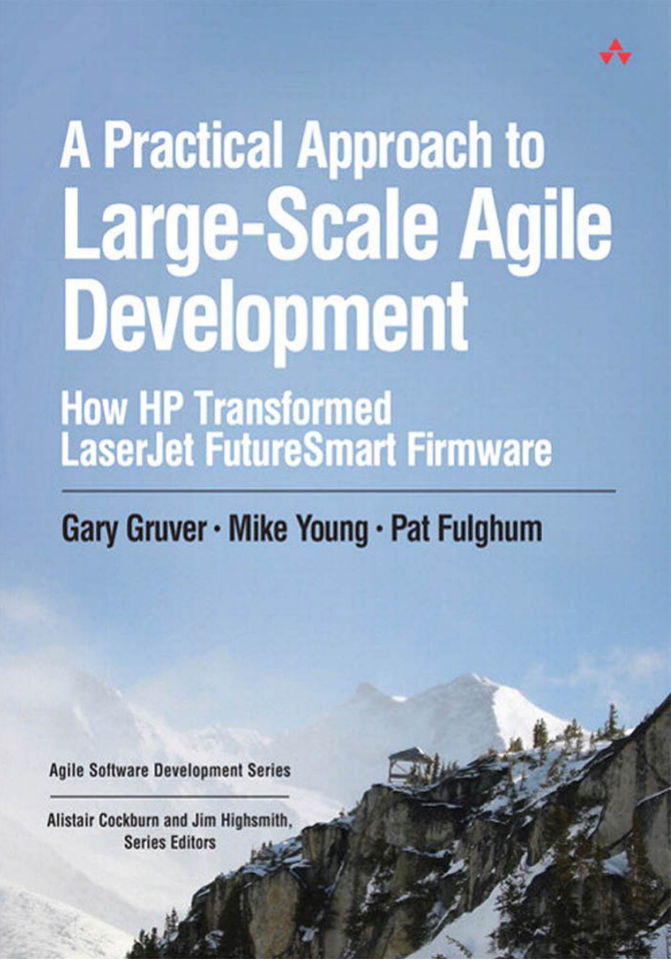

A Practical Approach to Large-Scale Agile Development

The Agile Software Development Series

Alistair Cockburn and Jim Highsmith, Series Editors

Addison-Wesley

Visit **informit.com/agileseries** for a complete list of available publications.

Agile software development centers on four values, which are identified in the Agile Alliance's Manifesto*:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The development of Agile software requires innovation and responsiveness, based on generating and sharing knowledge within a development team and with the customer. Agile software developers draw on the strengths of customers, users, and developers to find just enough process to balance quality and agility.

The books in The Agile Software Development Series focus on sharing the experiences of such Agile developers. Individual books address individual techniques (such as Use Cases), group techniques (such as collaborative decision making), and proven solutions to different problems from a variety of organizational cultures. The result is a core of Agile best practices that will enrich your experiences and improve your work.

A Practical Approach to Large-Scale Agile Development

How HP Transformed LaserJet FutureSmart Firmware

Gary Gruver
Mike Young
Pat Fulghum

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Gruver, Gary, 1962-
A practical approach to large-scale Agile development : how HP transformed laserjet futuresmart firmware / Gary Gruver, Mike Young, Pat Fulghum.
pages cm
Includes bibliographical references and index.
ISBN 978-0-321-82172-0 (pbk. : alk. paper)
1. Agile software development. 2. HP LaserJet printers. 3. Computer firmware. 4. Software engineering. I. Young, Mike, 1965- II. Fulghum, Pat, 1964- III. Title.
QA76.76.D47G794 2013
005.1--dc23

2012034286

ISBN-13: 978-0-321-82172-0
ISBN-10: 0-321-82172-6

Text printed in the United States on recycled paper at
RR Donnelley & Sons in Crawfordsville, Indiana.

Second Printing: January 2015

Editor-in-Chief
Mark Taub

Executive Editor
Chris Guzikowski

Senior Development Editor
Chris Zahn

Managing Editor
Kristy Hart

Project Editor
Lori Lyons

Copy Editor
Barbara Hacha

Senior Indexer
Cheryl Lenser

Proofreader
Sarah Kearns

Technical Reviewers
Davie Sweis
Lisa Shoop
Robert Bogetti

Editorial Assistant
Olivia Basegio

Cover Designer
Alan Clements

Compositor
Nonie Ratcliff

Dedicated to our amazing wives—Carolyn, Alison, and Tammee.

This page intentionally left blank

CONTENTS

This page intentionally left blank

FOREWORD

This is the book we've all been waiting for! The Hewlett Packard LaserJet FutureSmart Firmware program described in this book—to revamp LaserJet firmware across the entire printer product line—shows how agile fulfills its promised benefits on a large, complex product delivery. Furthermore, the book is written by a team who did it—executive, program manager, and architect—not consultants (like me). The HP FutureSmart Firmware program (multiple products, four-year time frame) succeeded at many levels: It significantly contributed to the future of the LaserJet printer line, it changed an organization's culture, it transformed investment from 95% "get the product out," 5% innovation, to an incredible 40% to innovation (customer differentiation, and so on), reduced development costs by 40%, and reduced cycle time from two months to one day (as they measured cycle time). This program was large (around 400 people), distributed (around the world), complex (firmware, multiple products supported, complete architectural revamp), and fast changing (printer market conditions). This book details the journey that led to these accomplishments.

First, the program changed how HP markets the LaserJet line of printers. The HP web site now describes capabilities such as: transformational innovation, adaptability as businesses change, consistency of applications across printers, centralized management of multiple printers, and the capability to download business applications like a cell phone. The FutureSmart Firmware program delivered on business innovation.

Second, the program changed an organization's culture—not just in development, but over time in product management. In one of the more significant accomplishments, and not an easy one according to the authors, they changed from having each printer product manager impose feature requests for individual products, to a programmatic system synchronized to a firmware delivery train with a program manager who consolidated backlogs and prioritized not just on individual product needs, but on product line needs.

How many organizations decry that they are spending 75% or 80% or even 95% of their development resources on keeping the current system running? They may talk about maintenance and enhancements, but the bottom line is that little is left over to invest in new innovation. That was the case with the firmware system when this program started. They were spending 95% of their resources just keeping up with day-to-day changes, with little left for investing in the future. In my experience, the investment and persistence required to turn these situations around is rare. The FutureSmart team invested in a new architecture, built automated testing and continuous integration infrastructure, and refactored or rewrote code. The authors describe the details of their remarkable turnaround—and the bumps and bruises they got along the way. Not only were they able to turn their investment profile around, they reduced overall development costs by 40% in the process.

There are other organizations who have gone through significant agile turnarounds, but none that I know of whose participants have documented their journey as well for the rest of us to learn from. Gary Gruver, Mike Young, and Pat Fulghum have contributed their journey in a complete and thoughtful way.

Jim Highsmith
Executive Consultant, ThoughtWorks, Venice, Florida

Preface

Agile is everywhere. Especially in software development circles, you can't turn around without hearing the term. Agile books. Agile conferences. Agile forums. Dig a bit deeper and all the agile verbiage starts to come out: Scrum, XP, burndown charts. There's even new large-scale agile terms and practices like "Scrum of Scrums" to deal with the complexities of making agile work beyond the typical seven to ten person team. Interesting that something that in name is so "quick to respond" has become quite steeped in theory and process.

This is a story. A story of an experiment about taking agile principles and applying them on a large scale for the re-architecture of a code base and the ongoing innovation and delivery of a multimillion dollar business. A story of listening to what was being said about agile and lean and taking hold of the core principles. But then we needed to look hard at our business situation and use the agile practices that made sense for us, while throwing many renowned agile practices away. We also had to invent new ways of applying agile for our environment because we found they made us more productive. We had to combine basic agile principles with the very real business constraints and the needs we were experiencing. The books and literature described a lot about small, co-located agile teams. Could we really apply agile to more than 400 embedded software (firmware) developers scattered across four states and three continents that reported to four different business units? We needed to completely re-architect a code base with millions of lines of code and a 25-year HP LaserJet Printer legacy of backward-compatibility features to match. We also needed to keep an eye toward a critical future of innovation where

firmware was not just the sliver of code to run the hardware, but the complex brains across many devices to enable a new ecosystem of innovation. And all this was occurring in one of the longest economic downturns in U.S. history where our resources were constrained.

This is what drove us to take a leap this large. We had started down the agile path a few times previously (both top-down and bottom-up efforts), all with incremental success. But being hit with so many disruptive forces all at once meant we had to do things significantly differently: to think about improving developer productivity by 10x; to think about putting together an architecture incrementally instead of big-bang; to think about supporting a business model of releasing products not just once, but re-introducing them again and again every quarter with additional innovation; and to think about making quality an everyday, every person, technology-driven concept to fundamentally change the equation of how developers do their job and how long product and feature development cycles need to take.

This is not an academic approach to the problem. This is a practical approach to large-scale agile development based on a four-year experience at HP. It's about how we chose where to focus based on our specific business priorities and needs, and about the missteps we took along the way—and how agile allows and even encourages learning and adjusting as you go. Often the only way to know what works is to find out what doesn't. It's about how our transformation enabled the productivity and agility of a small-team atmosphere within the scale of a large organization, and how it reduced coordination and meeting overhead by putting metrics monitoring at everyone's fingertips so that management could quickly remove roadblocks and keep development on track.

Our hope is that providing this case study will encourage others to start their journey to realizing the dramatic breakthroughs in productivity enabled by agile/lean approaches. It is not a textbook on agile, because lots of those are available. It is intended to be a simple read that can be quickly consumed by busy leaders of large organizations to help them understand what is possible, the potential breadth of the changes, and provide ideas on how to get started. It is also intended to be a catalyst that

can easily be shared across an organization considering an agile transformation to get everyone excited about the possibilities and start building momentum for their own personal journey. The book is not about defining the exact right way to do agile, because the agile techniques and approaches are just tools for the real objective, which is transforming your business and improving the effectiveness of your software and firmware development processes. This book lays out what we did and why. It also documents the dramatic improvements in productivity we realized through this transformation. We decreased our development costs per program by about 70% while increasing the capacity for innovation in products and solutions. This was a huge organizational change that provided significant benefits for our business. Our hope is that by sharing this story, you will realize that these breakthroughs are not only possible for your organization, but are probably required to remain competitive moving forward.

This book is also intended for business leaders or aspiring business leaders across industries. Moving forward, software is becoming important not just for technology companies but for every business in terms of how your organization adds value or controls costs. Therefore, having a basic understanding of some leading edge approaches to software development, including the potential impact that technical changes can have on a business, will become more and more important. This book will help show how a fundamentally different approach with a technology can have a dramatic impact not just on the development cost structure but also on the value proposition of a product line.

We start the book (Chapters 1 through 5) by focusing on the foundation of what should drive any agile initiative (especially in a large-scale organization):

- What are the core agile **principles** you want to follow? (Chapter 1)
- What are the **business drivers** you are trying to optimize for in your company? You can make agile work for making your business successful, and not become a slave to it as an end in itself. (Chapter 2)

- How to put in place a strong **architecture** to base everything on—and how to get it in place in an agile way. (Chapters 3 and 4)
- What kind of **culture and management style** do you want to create? This is a powerful part of being successful in large-scale agile. (Chapter 5)

Chapters 6 and 7 then delve into the real nuts and bolts of our processes that enabled us to drive to 10x productivity improvements—continuous integration, automated multilevel qualification, user story definition, and light-touch capacity prediction. We didn't hit any ideal state right out of the gate—we iterated throughout the experience, learning some painful lessons as we went. Agile is just as important in adapting processes as you go as it is in delivering products and features.

The next portion of the book (Chapters 8 to 11) touches on the challenges of making large-scale agile work within a broad, distributed organization, including how to do project management, how to organize teams (pros and cons), and how to work across cultures.

In Chapters 12 to 15, we capture the details of the enterprise-capable tools we used, capture the business results of our agile case study, touch on the new concept called *enterprise agile*, and describe how we interface beyond firmware/software to all other partners in the business. Then we discuss how different our approach was for scaling agile from what is typically done in industry.

Our journey has been very involved and has come a long way; all of this can be very overwhelming to try to roll out all at once, so we end with perhaps the most important chapter of all, Chapter 16, "Taking the First Step"—about how to get started and be successful without trying to solve it all at once.

Although this book contains observations and best practices from our experiences, it also presents ideas for applying the learning to other situations that need productivity improvements with a whole different business case. Come enjoy the journey as we relive it here. Your business is

different from ours. Don't get distracted with any perfect practices we'll throw at you (everything's a work in progress here!). But be guided by the power of agile principles applied to your business and its unique personality, opportunities, and constraints.

Acknowledgments

Although it would be great to list every person who has contributed, that would be impossible. We want to thank every HP FutureSmart Firmware developer, tester, and manager who has made a difference in making this initiative real. Thanks to each of you for your commitment, passion, ideas, and willingness to make this experiment into a broad and successful reality. The concepts and practices and results presented in this book are the result of every person involved in this transformation. It was an amazing journey and we appreciate all your contributions along the way. Without everyone working together and learning from each other, this transformation would not have been possible. THANKS!

We appreciate the support of Von Hansen, our VP and General Manager at HP, in helping to lead this transformation and encouraging this publication. He was instrumental in driving this significant breakthrough for HP and providing us with the support and encouragement we needed along the way. Without his support and initiative, neither the book nor the transformation would have been possible.

We also owe a special thanks to Jim Highsmith for recognizing the value of this manuscript early on and providing the support and encouragement necessary to make it a reality. He helped guide us through the publishing process and helped push to get us the support we needed. Jim also helped get us better linked into the agile community, for which we are very grateful. Without his guidance and support, this book would have never been possible.

We also would like to thank everyone who took to the time to review early versions of this manuscript and provide valuable feedback. It is a better product because of your help. Thanks to Jim Highsmith, Jez Humble, Troy Pearse, Ajay Gupta, Arun Dutta, Keith Moore, Luciano Rocha, Joe Longo, Frank Riskey, Kimon Papahadjopoulos, Steve Townsley, Michael Turner, and Phil Magnuson.

About the Authors

When we started this journey, we didn't have much background with agile. We were a team that had a lot of passion around transforming our development processes to improve our effectiveness. We were encouraged by what we read about agile and decided to leverage and apply it to our large-scale embedded firmware development at Hewlett-Packard. What is captured in this story is based on what we found to be effective in transforming our business. In hindsight, it probably would have made sense to read more and better leverage the agile community so we did not have to learn a lot of these lessons through the school of hard knocks. What we found as we got more involved with the agile community as we worked through the publishing process is that much of what we developed was also being implemented by others in the agile community. Because of budgeting constraints, we were not reading extensively or using consultants, so some of the terminology we use is not industry standard. During the review process, we decided that instead of changing all our terminology to match industry standards, we would keep what we developed but insert references to other books where readers could find the latest ideas to help accelerate their journey.

So why did we write this book? We've found that most agile books on the market are written by those who spend their time studying, lecturing, consulting, and writing about best practices from working with different businesses. We thought it might be helpful to publish a book based on how these different agile ideas came together in one company. Therefore, this is our story of turning a large-scale development organization into an agile machine over the past four years. It hasn't always been pretty or

easy, but every step along the way has been taken because it worked—in real life. We haven't achieved nirvana, but we've set up a pretty amazing system and environment that works well for us and HP's business needs.

We each have very different roles at HP, and we believe that having someone in each role we've played is critical to the success of an effort like this.

Gary Gruver is formerly the Director of Engineering for HP's LaserJet Core Firmware Lab, and he worked at HP for 22 years. He is currently VP of Release, QA, and Operations at macys.com. Any major initiative needs a true business sponsor—someone who has truly caught the vision of agile, and who can make the business and financial decisions necessary to get huge breakthroughs to happen. Gary has also been able to bring a "manage to metrics" approach that rallies everyone to common measurable objectives without requiring lots of meeting and coordination overhead. Of course, his most critical role is buying lunch during particularly busy sprints for anyone working weekends to finish off key features. His favorite hobbies are cycling and skiing with family (he's married with two daughters).

Mike Young is the program manager directing day-to-day efforts across our many distributed teams at HP's LaserJet Core Firmware Lab. Mike has been involved in development of HP LaserJet Printers for 18 years, and he previously designed satellite control systems for Hughes Aircraft Company. He also is one of the strongest advocates of agile approaches and helped get the organization started down this path before anyone really knew we were doing agile. His hobbies are family (he's married, with two daughters and two sons) and playing racquetball. In agile, we've found that a program manager should spend most of his/her time watching the metrics and quietly coordinating behind-the-scenes to cater to the bottleneck. In our sprint checkpoints, we tend to minimize slideware and maximize problem solving and demos of new user stories.

Pat Fulghum is architect of the HP LaserJet FutureSmart firmware and its development team's agile toolset. Pat's been at HP for 24 years. He found out during the past few years that his favorite escape is scuba diving in Maui with his family (he is married and has a son and a daughter).

A large-scale agile initiative requires a central architect who can help maintain architectural integrity amid many pressures to do otherwise (which keeps the system enabled for the future) and who has the vision for making sure the architecture supports both firmware development and qualification. Pat still loves to get in and dig deep to solve vexing technical challenges. He also loves to find developer productivity improvements (build time, triage time) and has been the passion behind our "10x productivity improvement" vision.

Chapter 2

Tuning Agile to Your Business Objectives

We started with the "why" of agile practices—that is, principles. But there's a deeper "why" that we need to explore. It's really the "why" of the principles you choose to follow. Although agile is a powerful concept, becoming agile just because "it's the thing to do" won't automatically help a business achieve what it is trying to accomplish. To successfully create the significant breakthroughs in your development effectiveness that are possible with agile, it has to be aligned with *why you want to do it* in the first place and *what you need to achieve from it*. You should be agile not just to be agile, but to drive the business results. Start by describing your business situation.

First, sit down and identify your current business realities (where the money and time is going) and strategic objectives (where the money would ideally be spent for your business situation):

- **Cost and cycle-time drivers**
  What are the activities that are consuming your resources and limiting your ability to deliver on time?
- **Value proposition**
  What are your products or services really trying to achieve for the customer?

The final step in establishing the backdrop for an agile transformation and making sure the efforts are tied to your real business needs is to combine the two lists (where are you investing now; where do you need to invest) for a clear view into the problem areas. Use this analysis to *establish clear development objectives* for your organization. *The biggest cost drivers that aren't key to the value proposition are targets for improvements.* If these cost drivers can be architected out of the system, automated, or engineered away, it can free up resources for innovations critical to the value proposition.

The best way to talk about how we tuned agile to our business objectives is to clearly spell out our business situation before our large-scale agile experience began. So in this chapter, we'll give you an overview of HP FutureSmart Firmware that we're using as the case study. We'll identify our costs and cycle-time drivers prior to our agile transformation, explain the value proposition our business needed, and then list the development objectives that came out of our analysis and would effectively close the gap we faced.

Background: HP FutureSmart Firmware Case Study

HP FutureSmart Firmware is the name the business uses to market the latest embedded code used to control LaserJet hardware and enable solutions resident on the device. A typical laser printer consists of the electromechanical print engine, which is controlled by a formatter. The formatter is made up of both electronics and logic. The logic is referred to as *firmware* but can be considered a full-on multitasking operating system. In this case of the firmware for enterprise-class printers and copiers (FutureSmart), it is as complex as the operating system and logic running your PC or laptop or smartphone.

Our business challenges started with a predicament of two-year long development cycles for delivering firmware, and of complex embedded software that had been slowly aging over many years and needed to be

re-architected. Big-bang integrations were frequent. Before learning about agile, we had some early improvements and got to the point of 8-week development cycles, a daily build or two, and a nightly smoke test. But even with these significant improvements, some significant inefficiencies existed.

Cost and Cycle-Time Drivers Prior to HP FutureSmart Firmware

The first step is to understand how resources are being deployed and what activities are driving development costs. Honestly assess where software development dollars are spent. It is also important to understand the cycle time for a developer to implement a change and then get feedback on if it works.

Throughout this experience, we've had around 400 developers worldwide needing to get firmware and test changes integrated into a firmware system consisting of several million lines of code, with very high quality expectations. When we started our transition to agile development, we had created a complex environment and code base over many years that took most of our efforts just to keep it going:

- Ten percent of our staffing was for "build bosses" (someone on each team designated as its full-time code integrator) plus a central integration team to accomplish the one or two builds per day we were doing, with many teams doing integration their own way. This was a very manual process of integrating and reverting code, consuming several highly qualified engineers who, as a result, spent very little time actually coding. In this environment, each project team would have a build boss that would gather all the changes by the team every few days and bundle them into a collection of changes. These changes would then be provided to the integration team that would be taking changes from 15 different build bosses for a nightly build with a smoke test. This nightly build would then be provided for additional testing over time.

This approach resulted in a resource sink, but more importantly, it could be up to a week from the time a developer made a change until it got into broader testing on the main code branch to see if it worked.

- Twenty percent of our resources were spent doing detailed planning for future feature commitments that quickly became obsolete or were never delivered. Business and marketing expected a clear "final list of features" one year before product introduction. To provide that commitment, we worked on detailed work breakdowns, schedules, integration plans, and estimates, all of which required constant maintenance and revision, because new discoveries and adjustments are an integral part of any high-tech research and development (R&D) effort.
- Twenty-five percent of our resources were consumed porting the existing codebase and features from one product to another. Because of schedule pressures, we hadn't spent sufficient time to abstract out the code and encapsulate product differences. We also ended up splitting the organization and creating three distinct branches of the previously common code. This meant more focus for each part of the business, but fewer resources for assuring code maintainability.
- Fifteen percent of our development costs were for manual test execution, which was a significant cost driver. Although we had a very large test suite, most of it needed to be executed by technicians, which was a large chunk of the budget. It also meant very long feedback loops from test to development. It was sometimes weeks or even months between when a firmware change was made and when a test actually found an issue. This made for long find/fix cycles, and it consumed a large chunk of the budget. This also meant that we frequently could not add products to our plans because we did not have the resources for testing them.
- Twenty-five percent of development resources were deployed supporting existing products, either fixing customer change requests or making sure we had a consistent set of features across printers

and multifunction products (MFPs). With a focus over many years on getting each product to market, we had created multiple code branches that all had to be maintained for the products in the field.

- This left us with limited capacity to focus on the value proposition and customer differentiation that would actually provide the business value needed for continued success.

So that gives a clear picture of where we were allocating our resources. If you add it all up, the firmware development cost drivers were 95% "get the basic product out" and just 5% adding innovation. That is exactly opposite of where the business needed us to be in order to be competitive in the marketplace. So where did we *want* to be spending our money? What was the business asking for? The next section describes our value proposition for making such a substantial change.

Value Proposition of Re-Architecting the HP FutureSmart Firmware and Processes

After establishing where you are spending your resources, it is important to clearly understand the value proposition of your product. Is your primary goal to reduce cost for a given functionality? Is it to release the largest number of products at a given cost? Or is the real opportunity to provide clear differentiation to the customer? Each of these is valid, and there are many more, but the decisions around the trade-offs to make in transitioning your development processes will be dramatically different depending on your specific business value proposition and cost drivers.

To start our agile change, we stepped back and asked what we really wanted to accomplish. What if we could change that "95% turn the crank" reality into something very different with innovation at our core? Following are the real business drivers that we established as our vision and value proposition:

- Our firmware had been on critical path for nearly every product delivered for more than 20 years. We sorely needed to get it off that critical path. How could we *deliver firmware early and often with even higher quality*?
- Because such a large percentage of our resources was spent on the "turn the crank" activities mentioned previously, a significant pent-up market demand existed for more features and innovation. For four years, we tried to spend our way out of the problem, increasing firmware R&D investment dollars by two and a half times across multiple versions of the code base that had split off in an attempt to let each business unit control its own destiny. But it didn't seem to help. We needed to *significantly improve developer productivity and organization agility* to truly lead the market in all desired product attributes and features. We needed to engineer a solution.
- Our business drivers were also changing. Customers had been moving from a previous focus on "buying up to the latest product for faster printing" to needing an *advanced and consistent set of Multi-Function Printer (MFP) features for workflows/solutions* that have the MFP as an integral part. Previously, it had been okay for different products to have different capabilities. But after MFPs became an integral part of their workflows, customers started demanding consistency in the feature set. The technology curve with printers was to the point where the hardware engine speeds and print quality were satisfying customers, and we didn't need to keep ramping up the curve of higher speed or print resolution. More and more product differentiation began originating in firmware. Our firmware had transitioned from a "thin layer of code to help control the print engine" to being more software-like as the critical enabler for supported workflows and solutions. Customers also began to manage their fleet of printing devices, raising the importance of managing devices in a consistent manner.

With these clear cost drivers and opportunities as our backdrop, we were prepared to put agile to work for us (not us for it) and tackle these difficult but critical disconnects in our investment versus value-add picture.

Establish Development Objectives from the Business Analysis

As we started out on our agile journey, we translated this business picture of "where we were spending our money" and "what the business needed" into a clear set of **firmware development objectives** that we felt would close the gap between where we were and where we needed to be. Our goal was that these objectives would help *unleash the product roadmap and enable innovation by reengineering the code and development processes*:

- Create a stable application code base that is always close to ready for release.
- Automate tests and run a full set of regression tests every night.
- Automate the integration process, including autoreverting any code that is not up to par.
- Significantly reduce the work needed to get new products working with high quality and out the door to the market.
- Re-architect to remove product differences, enabling one branch for all products (even refreshes of released products).
- Improve developer productivity by a factor of 10 (build times, streamlined processes).
- Create a common development environment so engineers can easily help across teams.
- Reset expectations and reduce feature estimation activities (commit by delivering).

There was a final consideration in determining how to go forward: Every team or business must not only step back and ask about resource allocation and value-add, but also about the capacity of the organization to absorb change. How much change can the organization handle, how fast, and how many ideas can be driven from your position in the organization? This example involves a large amount of change over a long time period with a big team. Agile transformation can only happen as quickly as your organization has the capacity to invest and is ready to embrace significant change along with that investment. It also matters how influential the thought leaders are in your organization. The final chapter goes much more into the best way to start, but no matter what, make sure you start with the items that will give the biggest bang for the buck and are appropriate for your organizational influence and leverage.

In the following chapters, we will share our experiences and changes that enabled the following results in hopes that it will inspire your organization to start your journey toward transforming your business:

- 2008 to present overall development costs reduced by 40%
- Number of programs under development increased by 140%
- Development costs per program down 78%
- Firmware resources now driving innovation increased by a factor of 8 (from 5% working on new features to 40%)

Summary

What are your business objectives and value proposition? How are you spending your time and resources? What are the pain points you want to overcome? Do your investment areas match your objectives? With our case study as an example, we encourage you to do the same exercise. This will allow you to create a vision for the organization and a roadmap of how to get there. It's easy to get lost in the day-to-day and sprint-to-sprint nature of agile. Having the vision and strategy for where you want to be in the medium or long term is a powerful backdrop to measure all activities against and know if you're being successful.

INDEX

Symbols

A

B

C

D

E

F

G–H

I

J–K

L

M

N–O

P

Q–R

S

T

U

V

W–Z